BRANCHEMENT CONDITIONNEL ET BOUCLES

Voir le cours en ligne pour plus de détails.

1 Rappels sur les expressions booléennes

```
type(False) # bool

x = 2

4 == 5       # False : teste l'égalité
x == 2       # True
x = 3        # affectation
x == 2       # False
x != (x+1)   # True
x < (x+1)    # True
x >= 4       # False
```

Pour tester une égalité, on utilise le double égal : `==`

2 Branchement conditionnel

En français	En Python

Si *condition*,
Alors *instructions si vrai*,
Sinon *instructions si faux*.
 	
suite du programme
	if `condition`:
	`    instructions si vrai,`
	else:
	`    instructions si faux.`
	`suite du programme`

```
if a >= 0:
  print("a est positif ou nul")
else:
  print("a est strictement négatif")

if a > 10:
  print("a est supérieur ou égal à 11")
elif a >=-10:  # sinon si
  print("a est compris entre -10 et 10")
else:
  print("a est inférieur ou égal à -11")

# intégré à une fonction :
def memeSigne(a,b):
  if a*b >= 0:
    return "a et b sont de même signe"
  else:
    return "a et b sont de signes opposés"
```

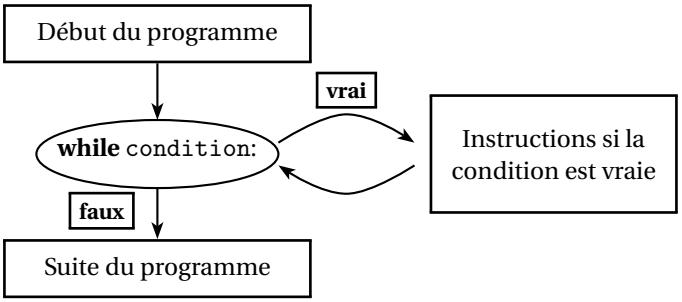
Remarque : Testez toujours vos fonctions sur plusieurs cas.

3 Boucle conditionnelle while

La boucle `while` consiste à réaliser des instructions tant qu'une assertion est vraie.

En français	En Python

Tant que *condition est vraie*,
Faire *instructions*,
Fin
suite du programme
	while `condition`:
	`    bloc d'instructions,`
	
	`suite du programme`




```
# compte de 1 à 10
i = 1
while i <= 10:
  print(i)
  i += 1
print('fini !')
```

4 Boucle for

Syntaxe	Signification
`range(`n`)`	entiers de 0 à $n-1$
`range(`p`,`n`)`	entiers de p à $n-1$ (1)
`range(`p`,`n`,`k`)`	entiers de p à $n-1$, avec un pas de k (2)

(1) Si $p \geq n$, alors `range` est vide (mais ne renvoie pas d'erreurs). p et n peuvent être négatifs.

(2) k peut être négatif, pour un parcours décroissant.

En français	En Python

Pour i parcourant *l'intervalle...*,
Faire *instructions*,
Fin
suite du programme
	for i in range`( ...):`
	`    bloc d'instructions,`
	
	`suite du programme`

Remarque : **for** peut parcourir autre chose qu'un intervalle : prendre les valeurs successives d'une liste par exemple.

```
# compte de 1 à 10
for i in range(1,11):
    print(i)
```

5 Erreurs récurrentes

- ne pas oublier les ":" avant de commencer un bloc
- penser à indenter les blocs
- pour tester une égalité, mettre un "==".
- **if** n'a pas de majuscule.
- **while**
  - Ne pas oublier d'**initialiser** le compteur **avant** la boucle.
  - Ne pas oublier d'**incrémenter** le compteur **dans** la boucle.